Cấu trúc dữ liệu theo sơ đồ lớp UML

Nội dung

- ✓ Giới thiệu ngôn ngữ mô hình hóa thống nhất UML
- ✓ Các đối tượng và sự kết hợp
- ✓ Tổng quát hóa
- ✓ Sơ đồ chuyển trạng thái

UML

- Ngôn ngữ mô hình hóa thống nhất (Unified Modeling Language - UML)
- UML là ngôn ngữ để:
  - trực quan hóa (visualizing)
  - đặc tả (specifying)
  - xây dựng (constructing)
  - tài liệu hóa (documenting)

các cấu phần (artifact) của một hệ thống thông tin

Khái niệm lớp trong mô hình UML khá tương tự với khái niệm thực thể trong mô hình thực thể-liên kết. UML là một trong những công cụ cơ bản để mô tả cấu trúc hóa dữ liệu.

UML

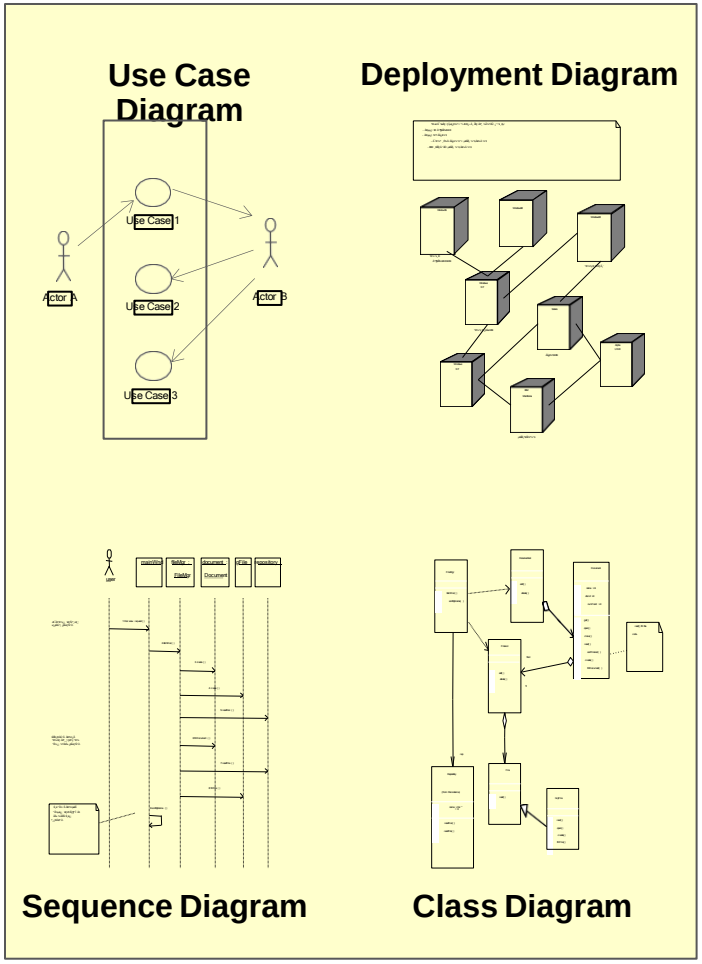
- UML là ngôn ngữ trực quan
- Giúp công việc phát triển được xử lý nhất quán, giảm thiểu lỗi xảy ra
  - Giúp dễ hình dung hơn cấu trúc của hệ thống
  - Hiệu quả hơn trong việc liên lạc, trao đổi
    + Trong tổ chức
    + Bên ngoài tổ chức

UML

- Các mô hình UML có thể kết nối trực tiếp với rất nhiều ngôn ngữ lập trình.
  - Ánh xạ sang Java, C++, Visual Basic…
  - Các bảng trong RDBMS hoặc kho lưu trữ trong OODBMS
  - Cho phép các kỹ nghệ xuôi (chuyển UML thành mã nguồn)
  - Cho phép kỹ nghệ ngược (xây dựng mô hình hệ thống từ mã nguồn)

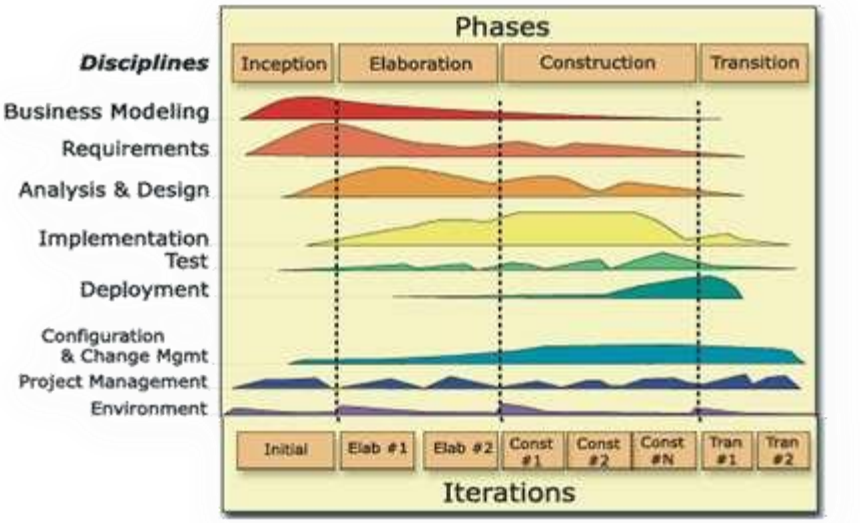
UML

- UML là ngôn ngữ tài liệu hóa
- Tài liệu hóa kiến trúc, yêu cầu, kiểm thử, lập kế hoạch dự án, và quản lý việc bàn giao phần mềm
- Các biểu đồ khác nhau, các ghi chú, ràng buộc được đặc tả trong tài liệu



UML

- UML là ký pháp chứ không phải là phương pháp
  - UML có thể áp dụng cho tất cả các pha của quy trình phát triển phần mềm
  - "Rational Unified Process" - quy trình phát triển cho UML



Lịch sử phát triển

- Vào 1994, có hơn 50 phương pháp mô hình hóa hướng đối tượng:
  - Fusion, Shlaer-Mellor, ROOM, Class-Relation,Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS …
- "Meta-models" tương đồng với nhau
- Các ký pháp đồ họa khác nhau
- Quy trình khác nhau hoặc không rõ ràng

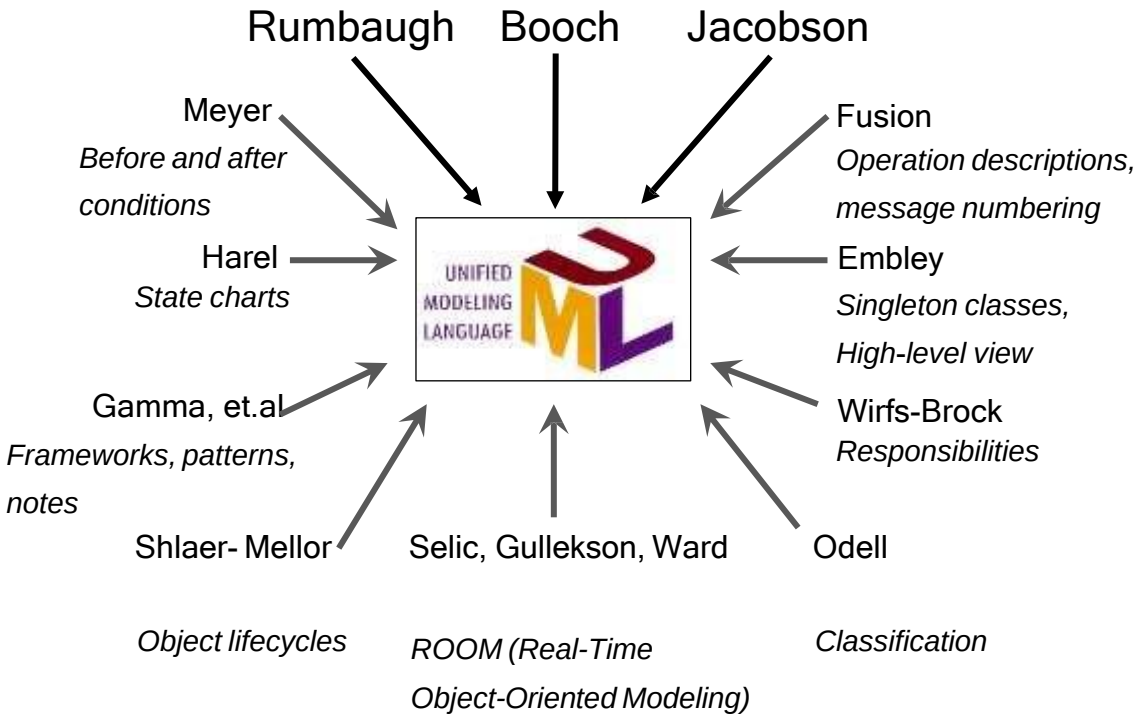
→ Cần chuẩn hóa và thống nhất các phương pháp

Lịch sử phát triển của UML

- UML được 3 chuyên gia hướng đối tượng hợp nhất các kỹ thuật của họ vào năm 1994:
  - Booch91 (Grady Booch): Conception, Architecture
  - OOSE (Ivar Jacobson): Use cases
  - OMT (Jim Rumbaugh): Analysis
- Thiết lập một phương thức thống nhất để xây dựng và "vẽ" ra các yêu cầu và thiết kế hướng đối tượng trong quá trình PTTK phần mềm → UML được công nhận là chuẩn chung vào năm 1997.

Lịch sử phát triển của UML

UML là ngôn ngữ hợp nhất các mô hình khác nhau



Mục đích

- Chuyển các yêu cầu của bài toán thành một bản thiết kế của hệ thống sẽ được xây dựng
- Tập trung vào quá trình phân tích các YÊU CẦU của hệ thống và thiết kế các MÔ HÌNH cho hệ thống đó trước giai đoạn lập trình
- Được thực hiện nhằm đảm bảo mục đích và yêu cầu của hệ thống được ghi lại một cách hợp lý trước khi hệ thống được xây dựng
- Cung cấp cho người dùng, khách hàng, kỹ sư phân tích, thiết kế nhiều cái nhìn khác nhau về cùng một hệ thống

Các công cụ UML

- Công cụ mã nguồn mở:
  - EclipseUML
  - UmlDesigner
  - StarUML
  - Argo UML...
- Công cụ thương mại:
  - Enterprise Architect
  - IBM Rational Software Architect
  - Microsoft Visio
  - Visual Paradigm for UML
  - SmartDraw...

Các biểu đồ UML

- Biểu đồ lớp (Class Diagram)
- Biểu đồ use case (Use Case Diagram)
- Biểu đồ hoạt động (Activity Diagram)
- Biểu đồ tương tác (Interaction Diagrams)
  - Biểu đồ trình tự (Sequence Diagram)
  - Biểu đồ giao tiếp/cộng tác (Communication/Collaboration Diagram)
- Biểu đồ trạng thái (Statechart Diagram)
- Biểu đồ cấu trúc tĩnh (Static Structure Diagrams)
  - Biểu đồ lớp (Class Diagram)
  - Biểu đồ đối tượng (Object Diagram)
- Biểu đồ thực thi (Implementation Diagrams)
  - Biểu đồ thành phần (Component Diagram)
  - Biểu đồ triển khai (Deployment Diagram)

Biểu đồ lớp

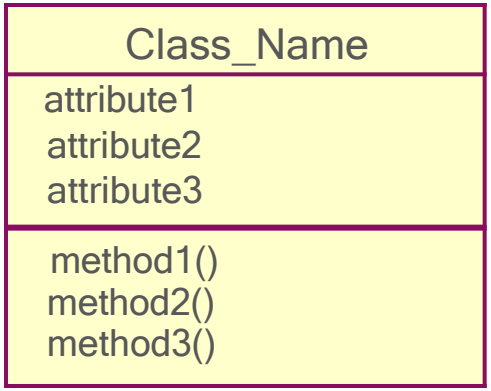
Class diagram

Biểu đồ lớp

- Biểu đồ lớp (Class diagram – CD) chỉ ra sự tồn tại của các lớp và mối quan hệ giữa chúng trong bản thiết kế logic của một hệ thống
  - Chỉ ra cấu trúc tĩnh của mô hình như lớp, cấu trúc bên trong của chúng và mối quan hệ với các lớp khác.
  - Chỉ ra tất cả hoặc một phần cấu trúc lớp của một hệ thống.
  - Không đưa ra các thông tin tạm thời.
- Khung nhìn tĩnh của một hệ thống chủ yếu hỗ trợ các yêu cầu chức năng của hệ thống.

Lớp

- Sử dụng hình chữ nhật gồm 3 thành phần
  - Tên lớp
  - Các thuộc tính
  - Các phương thức



Biểu diễn thuộc tính

- Chỉ ra tên, kiểu và giá trị mặc định nếu có
  - `attributeName : Type = Default`
- Tuân theo quy ước đặt tên của ngôn ngữ cài đặt và của dự án.
- Kiểu (type) nên là kiểu dữ liệu cơ bản trong ngôn ngữ thực thi
  - Kiểu dữ liệu có sẵn, kiểu dữ liệu người dùng định nghĩa, hoặc lớp tự định nghĩa.

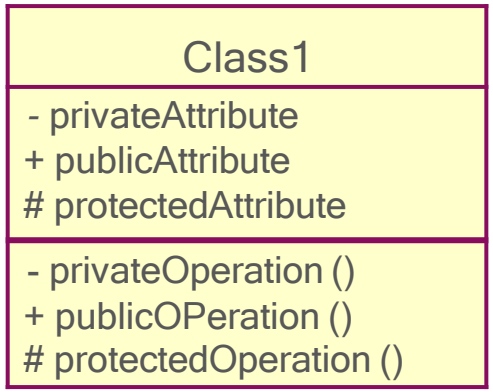
Mô tả phương thức

- Tên phương thức:
  - Mô tả kết quả
  - Sử dụng góc nhìn của đối tượng khách (client – đối tượng gọi)
  - Nhất quán giữa các lớp
- Chữ ký của phương thức:
  - `operationName([direction] parameter:class,...):returnType`
    + Direction: in (mặc định), out hoặc inout

Phạm vi truy cập

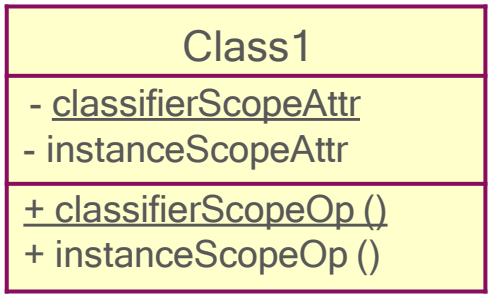
- Phạm vi truy cập được sử dụng để thực hiện khả năng đóng gói
- Sử dụng các ký hiệu

–	+	Public access
–	#	Protected access
–	-	Private access



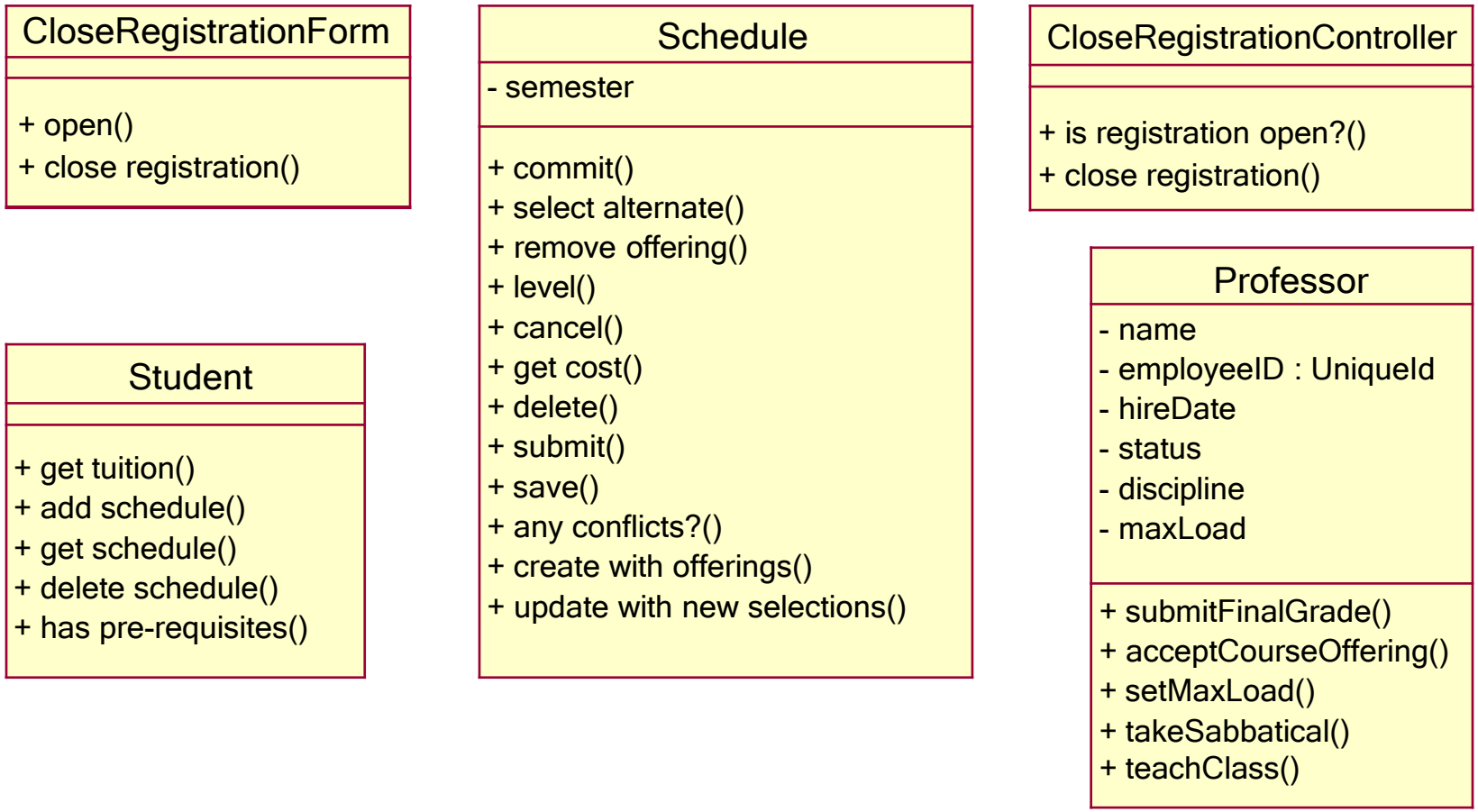
Thành viên lớp

- Phạm vi truy cập xác định số lượng thể hiện của thuộc tính/thao tác:
  - Instance (Thành viên đối tượng): Một thể hiện cho mỗi thể hiện của mỗi lớp
  - Classifier (Thành viên lớp): Một thể hiện cho tất cả các thể hiện của lớp
- Thành viên lớp (thành viên tĩnh) được ký hiệu bằng cách gạch dưới tên thuộc tính/thao tác.



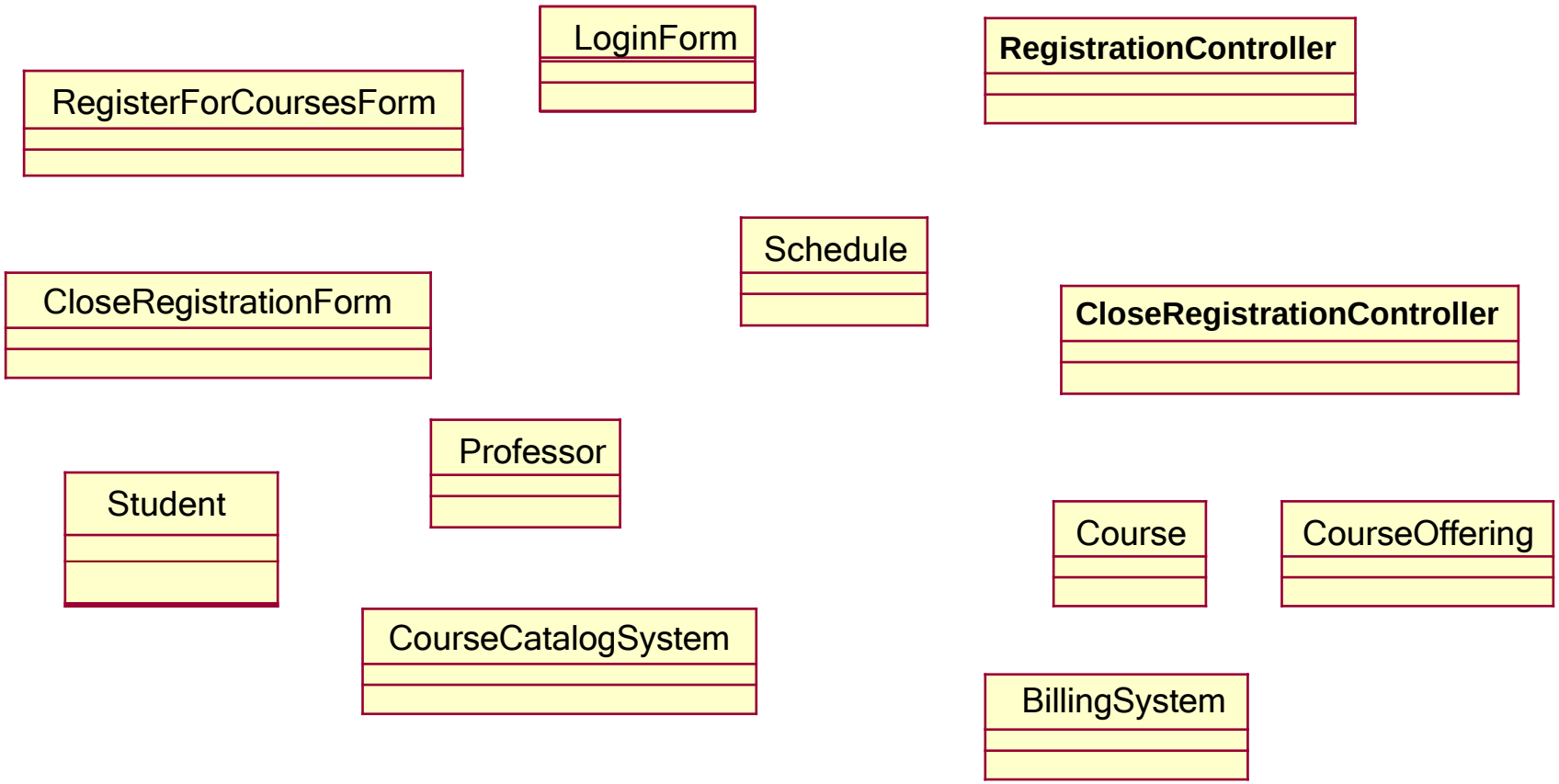
Ví dụ

- Hệ thống đăng ký khóa học



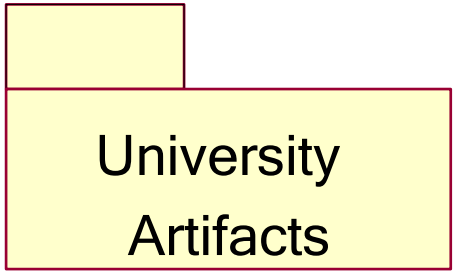
Ví dụ

- Biểu đồ lớp sơ lược

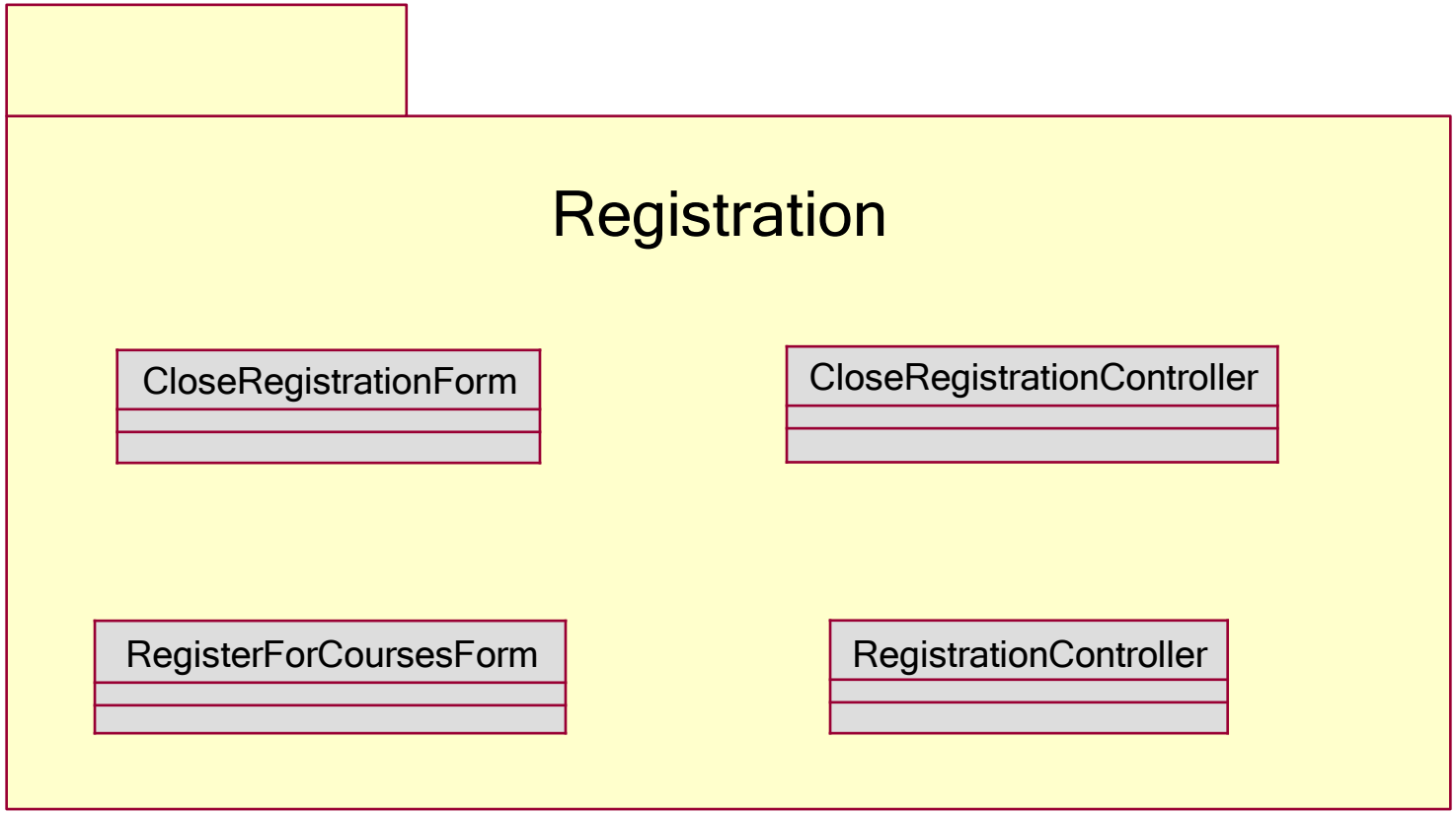


Gói

- Một cơ chế chung để tổ chức các phần tử thành nhóm.
- Một phần tử trong mô hình có thể chứa các phần tử khác.

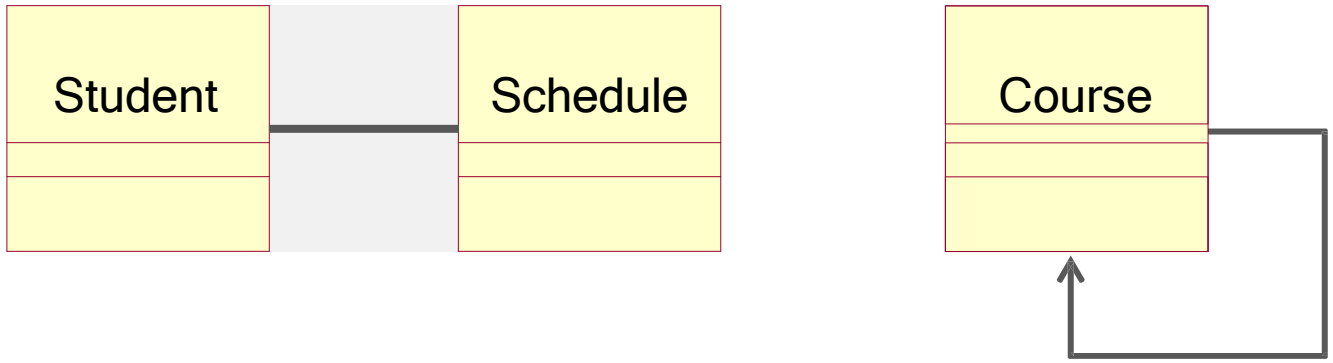


Ví dụ


Registration
CloseRegistrationForm
CloseRegistrationController
RegisterForCoursesForm
RegistrationController

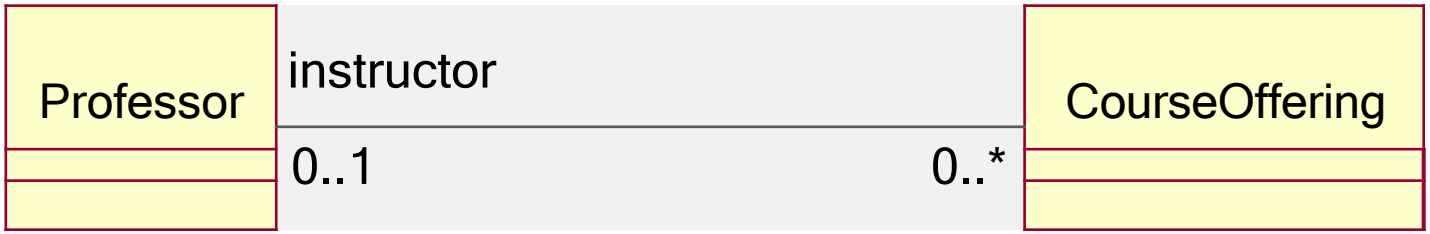
Liên kết

- Mối liên hệ ngữ nghĩa giữa hai hay nhiều lớp chỉ ra sự liên kết giữa các thể hiện của chúng
- Mối quan hệ về mặt cấu trúc chỉ ra các đối tượng của lớp này có kết nối với các đối tượng của lớp khác.

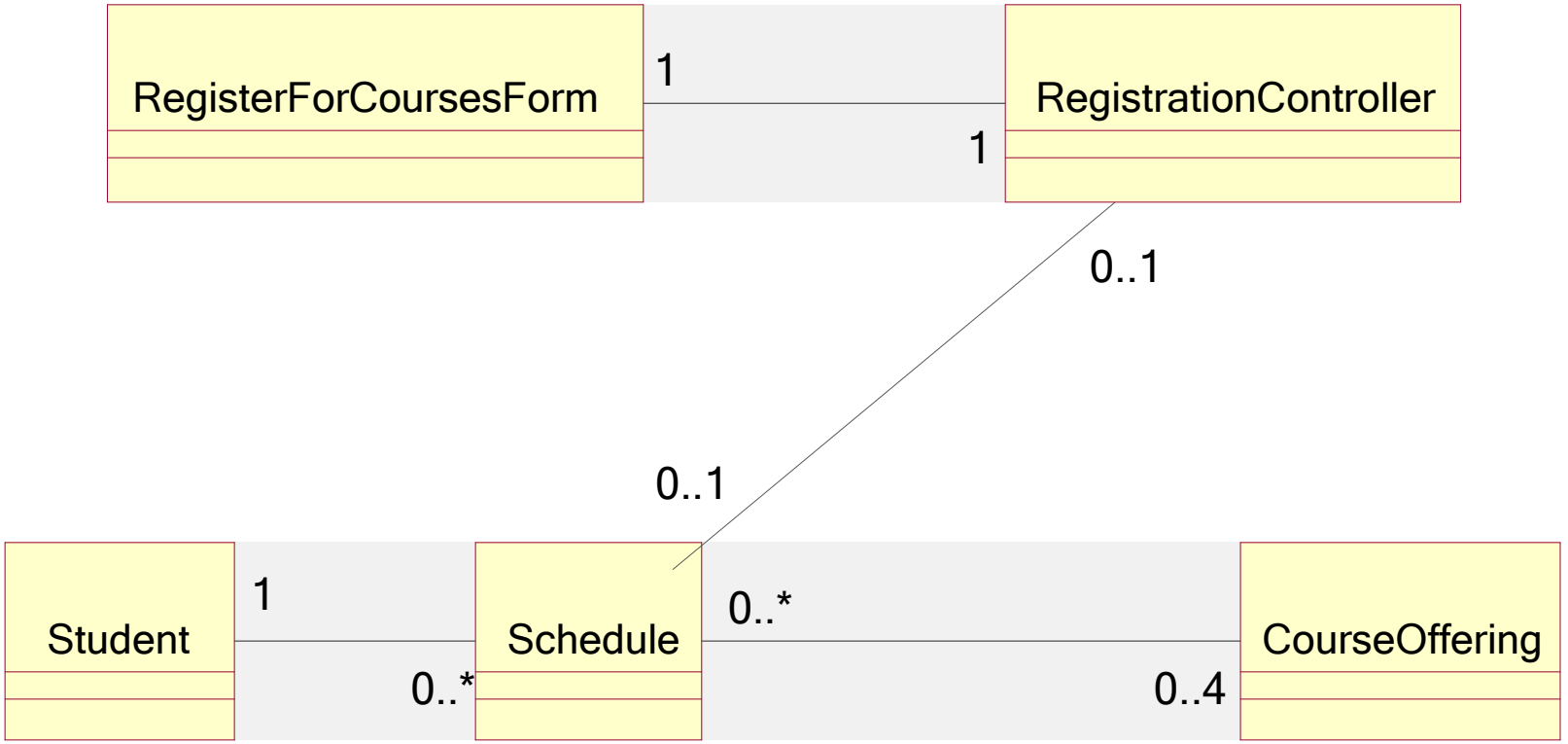


Bội số quan hệ

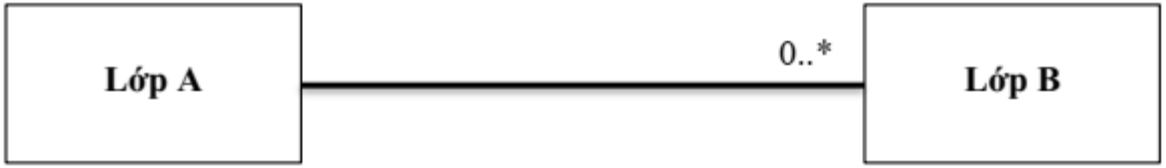
- Bội số quan hệ là số lượng thể hiện của một lớp liên quan tới MỘT thể hiện của lớp khác.
- Với mỗi liên kết, có hai bội số quan hệ cho hai đầu của liên kết.
  - Với mỗi đối tượng của Professor, có nhiều Course Offerings có thể được dạy.
  - Với mỗi đối tượng của Course Offering, có thể có 1 hoặc 0 Professor giảng dạy.



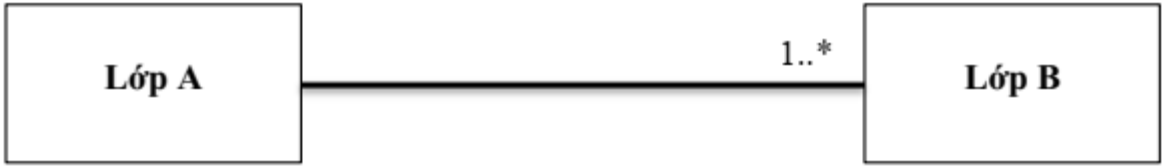
Ví dụ


RegisterForCoursesForm
1
1
RegistrationController
0..1
0..1
Student
1
0..*
Schedule
0..*
0..4
CourseOffering

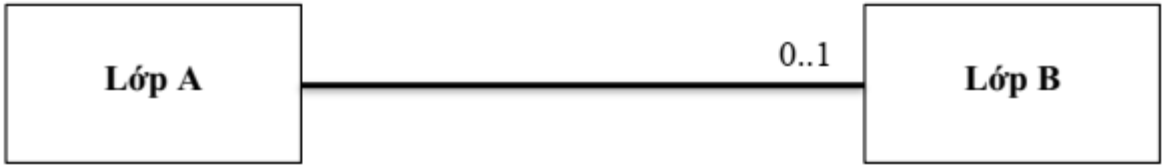
Ví dụ



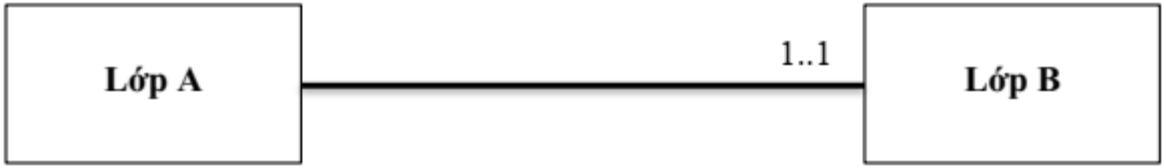
Một đối tượng của A *có thể* quan hệ với không hoặc nhiều đối tượng của B



Một đối tượng của A *luôn* quan hệ với một hoặc nhiều đối tượng của B



Một đối tượng của A *có thể* quan hệ với một thể hiện của B

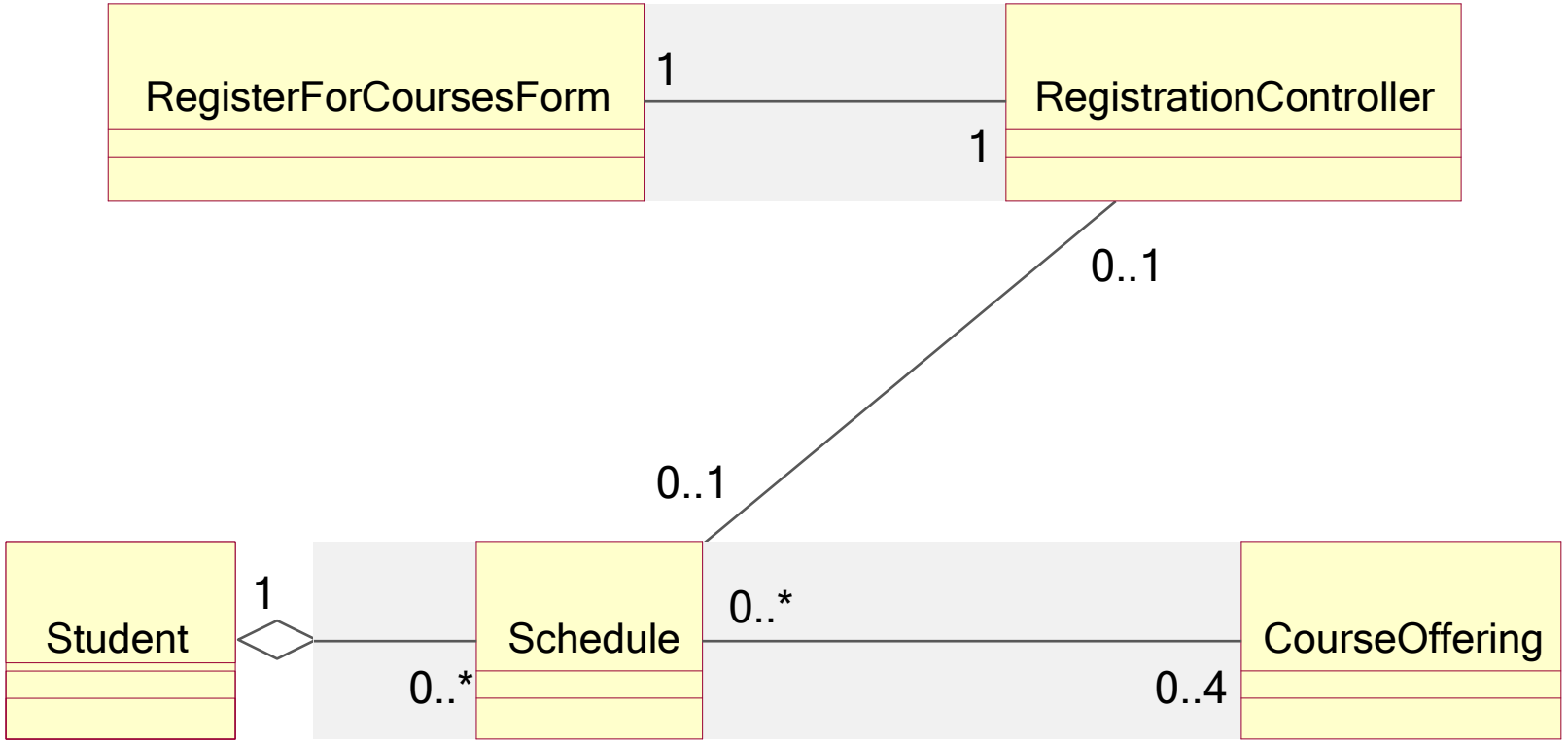


Một đối tượng của A *luôn* quan hệ với một đối tượng của B

Kết tập

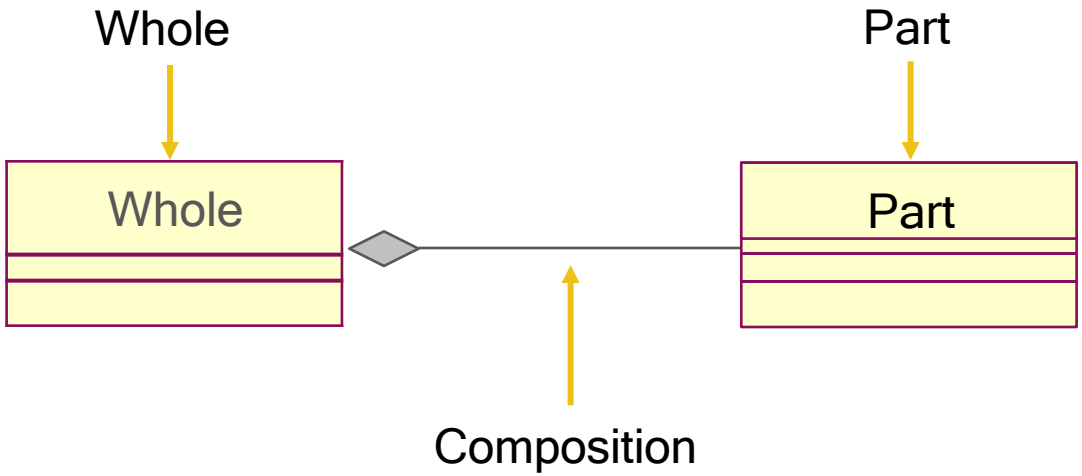
- Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.
  - Kết tập là mối quan hệ "là một phần" ("is a part-of").
- Bội số quan hệ được biểu diễn giống như các liên kết khác

Ví dụ


RegisterForCoursesForm
1
1
RegistrationController
0..1
0..1
Student
1
0..*
Schedule
0..*
0..4
CourseOffering

Cấu thành

- Một dạng của kết tập với quyền sở hữu mạnh và các vòng đời trùng khớp giữa hai lớp
  - Whole sở hữu Part, tạo và hủy Part.
  - Part bị bỏ đi khi Whole bị bỏ, Part không thể tồn tại nếu Whole không tồn tại.



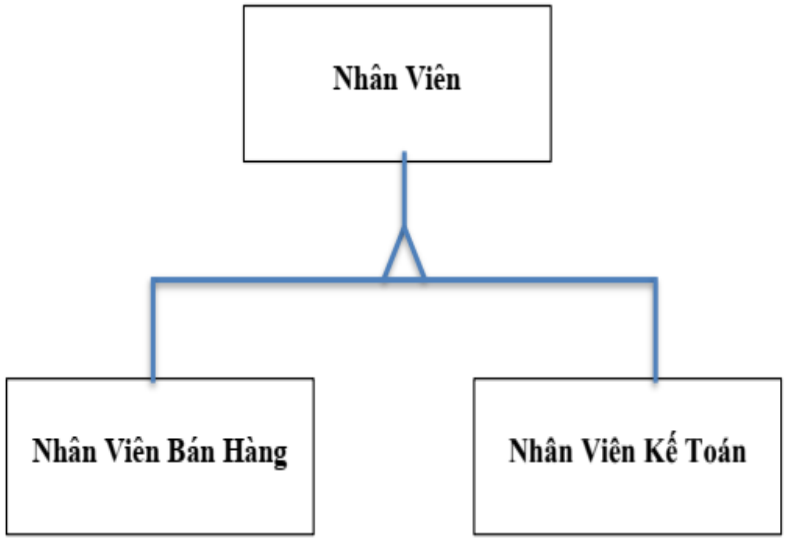
So sánh kết tập và cấu thành

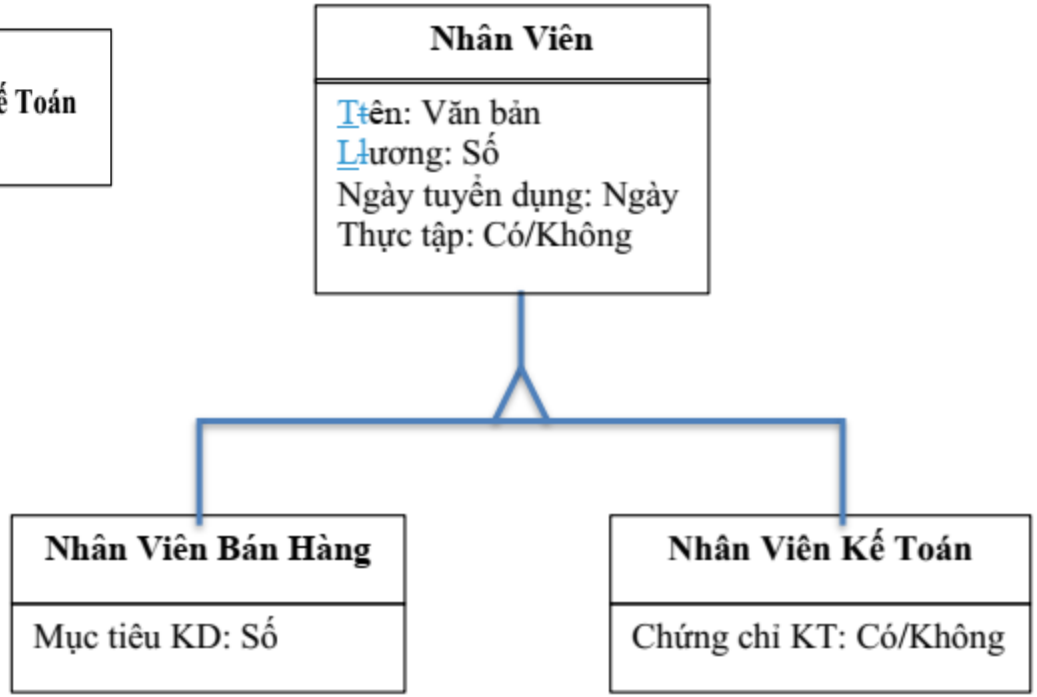
- Aggregation – University and Chancellor
  - Nếu không có trường Đại học (University), hiệu trưởng (Chancellor) không thể tồn tại.
  - Nếu không có Chancellor, University vẫn có thể tồn tại
- Composition – University and Faculty
  - University không thể tồn tại nếu không có các khoa (Faculty) và ngược lại (share time-life)
    + Thời gian sống của University gắn chặt với thời gian sống của Faculty
    + Nếu Faculties được giải phóng thì University không thể tồn tại và ngược lại

Tổng quát hóa (Generalization)

- Trong quá trình mô hình hóa, trong nhiều trường hợp các thực thể có nhiều thuộc tính chung giống nhau.
- Ví dụ nhân viên kinh doanh và nhân viên kế toán đều là các nhân viên nên sẽ có một số thuộc tính chung như tên, tuổi, ngày sinh v.v. Ngoài các thuộc tính chung giống nhau, các thực thể còn có các thuộc tính riêng khác nhau.
- Kỹ thuật tổng quát hóa của UML cho phép mô hình hóa hiệu quả các trường hợp trên.
- Kỹ thuật tổng quát hóa sắp xếp các lớp vào một cây ở đó các lớp tổng quát hơn sắp xếp ở trên và các lớp cụ thể được sắp xếp ở bên dưới.

Tổng quát hóa (Generalization)





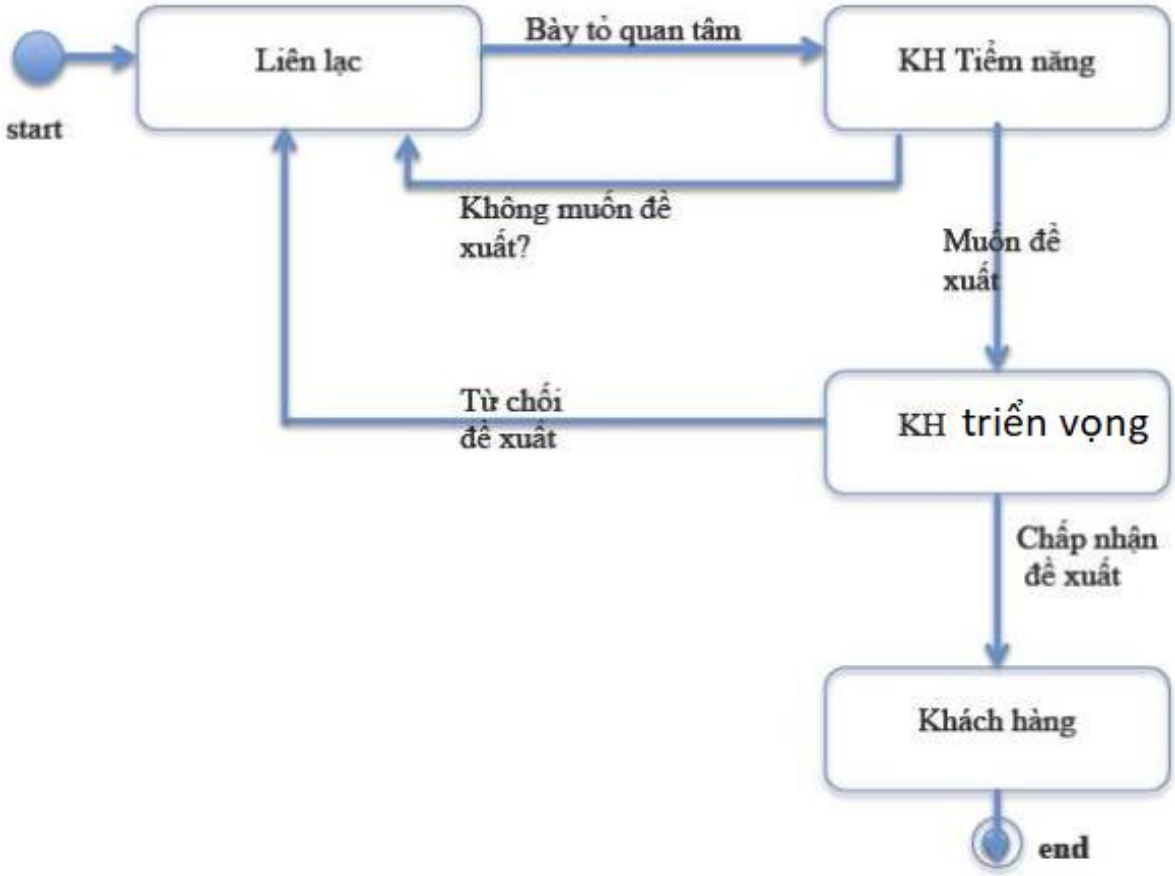
Tổng quát hóa (Generalization)

- Mối quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hoặc nhiều lớp khác
- Xác định sự phân cấp về mức độ trừu tượng hóa trong đó lớp con kế thừa từ một hoặc nhiều lớp cha
  - Đơn kế thừa (Single inheritance)
  - Đa kế thừa (Multiple inheritance)
- Là mối liên hệ "là một loại" ("is a kind of")

Sơ đồ dịch chuyển trạng thái

- Các thông tin về vòng đời của các thực thể đóng vai trò quan trọng trong thiết kế hướng đối tượng.
- Ví dụ, trong quy trình bán hàng, một khách hàng thường trải qua một số giai đoạn xác định từ xác định khách hàng tiềm năng, đến khách hàng triển vọng sau đó đến khách hàng thực sự.
- Người quản lý bán hàng phải quan tâm đến dịch chuyển trạng thái, ví dụ như bao nhiêu khách hàng triển vọng đã trở thành khách hàng thực sự, để có chính sách tiếp cận khách hàng phù hợp.

Sơ đồ dịch chuyển trạng thái



Sơ đồ dịch chuyển trạng thái

- Sơ đồ dịch chuyển trạng thái bổ trợ cho sơ đồ lớp bằng cách cung cấp thông tin về hành vi động của lớp.
- Các sơ đồ dịch chuyển trạng thái có thể được sử dụng để diễn đạt một cách chi tiết hơn các trạng thái khác nhau mà một lớp có thể dịch chuyển qua.
- Có trường hợp lớp không dịch chuyển qua trạng thái nào cả, trong trường hợp đó không cần sơ đồ dịch chuyển trạng thái.

XIN CẢM ƠN!